

LAMBDA STATION: ON-DEMAND FLOW BASED ROUTING FOR DATA INTENSIVE GRID APPLICATIONS OVER MULTITOPOLOGY NETWORKS.

A. Bobyshev , M. Crawford , P. DeMar, V. Grigaliunas, M. Grigoriev, A. Moibenko,
D. Petravick, R. Rechenmacher, FNAL, Batavia, IL 60510, USA

H. Newman, J. Bunn, F. Van Lingen, D. Nae, S. Ravot, C. Steenberg, X. Su, M. Thomas,
Y. Xia , California Institute of Technology, Pasadena, CA 91125, U.S.A.

Abstract

Lambda Station is an ongoing project of Fermi National Accelerator Laboratory and the California Institute of Technology. The goal of this project is to design, develop and deploy network services for path selection, admission control and flow based forwarding of traffic among data-intensive Grid applications such as are used in High Energy Physics and other communities. Lambda Station deals with the last-mile problem in local area networks, connecting production clusters through a rich array of wide area networks. Selective forwarding of traffic is controlled dynamically at the demand of applications.

This paper introduces the motivation of this project, design principles and current status. Integration of Lambda Station client API with the essential Grid middleware such as the dCache/SRM Storage Resource Manager is also described. Finally, the results of applying Lambda Station services to development and production clusters at Fermilab and Caltech over advanced networks such as DOE's UltraScience Net and NSF's UltraLight is covered.

PROJECT OVERVIEW

The main goal of Lambda Station project is to design, develop and deploy a network path selection service to interface production storage and computing facilities with advanced research networks. In the future, when corresponding API are available Lambda Station will also take on the task of negotiating with reservation or provisioning systems that may regulate the WAN control planes.

Policy based routing (PBR) is used to implement flow-specific routing in the LAN and at the border between LAN and WAN. In the next section of this paper we will discuss how Lambda Station serves the unprecedented

demands for data movement by running experiments such as CDF, D0, and BaBar as well as upcoming LHC experiments. From our point of view, available data communication technology will not be able to satisfy these demands simply by increasing bandwidth in LANs and commodity WANs due to technology limitations and high deployment and operational costs. Selective forwarding on per flow basis to alternate network paths is desirable for high impact data while leaving other traffic on regular paths. The ability to selectively forward traffic requires developing a control unit that is able to dynamically reconfigure forwarding of specific flows within local production-use routers on demand of applications. We refer to such a control unit as Lambda Station. If one envisions the optical network paths provided by advanced optical-based research networks as high bandwidth data railways, then Lambda Station would functionally be the railroad terminal that regulates which flows at the local site get directed onto the high bandwidth data railways. Lambda Station coordinates network path availability, scheduling, and setup, directs appropriate forwarding within the local network infrastructure, and provides the application with the necessary information to utilize the high bandwidth path. Having created Lambda Station, we introduce awareness and exploitation of advanced networking into data management services of our experiments. Figure 1 illustrates this main idea of the project. To fulfill its main goal the following parts of the project can be emphasized:

- Building a Wide Area testbed infrastructure
- Developing Lambda Station software, network aware applications, adapting production-use mass storage systems, running full-scale Scientific Discovery through Advanced Computation (SciDAC) applications to exploit advanced research networks

- Researching the behaviour of network aware applications with flow-based path selection

MOTIVATION OF THE PROJECT

The SciDAC Particle Physics Data Grid Collaboratory Pilot (PPDG) project develops, acquires and delivers vitally needed Grid-enabled tools for data-intensive requirements of these experiments. To fully exploit the science potential latent in their data, CDF and D0 at Fermilab and BaBar at SLAC are expanding their data

Another important use for very high throughput networks is to move the LHC data across the Atlantic from CERN in Geneva, Switzerland, to the U.S. Tier-1 regional centres: Fermilab for the CMS experiment and Brookhaven for ATLAS. From there data will be distributed to Tier-2 regional centres at universities like Caltech and UCSD. These data transfer facilities will have components of a quasi-real-time system as data taken at the LHC will have to be continuously distributed to the regional centers. Data streams of raw and reconstructed data ready for analysis are being spread

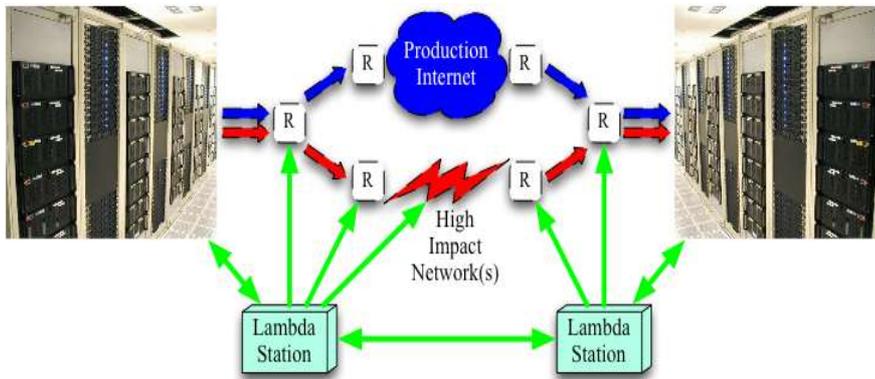


Figure 1: Lambda Station in control of traffic path

analysis to integrated distributed systems based on Grids. Moreover, U.S. physicists preparing for the analysis of data from the CMS and ATLAS detectors at the Large Hadron Collider (LHC) at CERN face unprecedented challenges:

- massive, globally distributed datasets growing to the 100 petabyte level by 2010
- petaflops of distributed computing
- collaborative data analysis by global communities of thousands of scientists.

PPDG, together with the NSF-funded iVDGL and GriPhyN projects, is moving to the development of next generation integrated Grid systems to meet these challenges, and to fully exploit the LHC's potential for physics discoveries. Today, all these high energy physics PPDG experiments' grid systems are limited by their treatment of the network as an external, passive, and largely unmanaged resource. Moreover, to date, no advanced network linking the U.S. HEP Laboratories and key universities involved in Grid and network development has been available to research and prototype solutions to these limitations.

over the distributed regional centers, selected and targeted to specific physics interests, to ensure full data access for U.S. physicists to LHC data and to serve analysis hot spots making data available to specific regional centers.

While the LHC model assumes logically dedicated 10Gb links between Tier0(CERN) and Tier1, Tier2 centers are not dedicated to LHC-only experiments. These sites, such as Fermilab are involved in several different scientific programs and need to have a mechanism capable of steering high impact LHC related traffic across the campus network, and on to available high bandwidth alternate paths.

To ensure full connectivity of the U.S. to CERN and full access of U.S. scientists to LHC data, the U.S. LHC software and computing efforts have started to put up U.S. LHC Edge Computing elements at CERN with sufficient data caching and data selection resources and 10Gbit connectivity from these systems across the Atlantic to the DOE funded link to CERN in Chicago. At both endpoints clusters of CPUs and storage elements are being used that are similar to the systems described

above. LHC data taking will start in 2007, and the LHC experiments are conducting a program of work to scale up to the required throughputs and functionalities that employs yearly “data challenges” that exercise the emerging end-to-end data flow systems to increasing degrees of complexity and size of data volumes.

Over the past several years, there has been a great deal of research effort and funding put into the deployment of optical advanced research networks, such as National Lambda Rail, CANet4, Netherlight, UKLight, and most recently, the DOE UltraScience Net. These networks potentially have the capacity and capabilities to meet the data movement requirements of the particle physics collaborations. To date, the focus of research efforts in the advanced network area have been primarily to provision, dynamically configure and control, and monitor the wide area optical network infrastructure itself. Application use of these facilities has been largely limited to demonstrations using test stands or small numbers of expensive high performance computing systems. The issue of integrating existing production computing facilities on production local network infrastructure with advanced, high bandwidth research networks is now beginning to be addressed. Fundamentally, this is a “last mile” problem between HEP production-scale computing facilities and the advanced networks. Lambda Station project is aimed at taking the first steps to address these issues.

LAMBDA STATION TESTBED

Building a WAN testbed for the Lambda Station project is a challenging task itself. Such a testbed should include components of the production infrastructures, both at the network site and computing and storage servers. At this time two HEP sites, Fermilab and Caltech, are involved in our testbed which is built around UltraScience Net (USN) and UltraLight (UL). At each site there are several test servers with 10Gb/s connections, storage clusters of “white box” nodes with 1Gb/s connections, a Lambda Station server, as well as a production LAN. The topology of the testbed is depicted in Figure 2. The Lambda Station at each site is allowed and able to reconfigure production routers on its own site to steer traffic of test or production clusters onto USN or UL instead of the standard ESNET path.

LAMBDA STATION SOFTWARE

An overview of Lambda Station's design and software was presented in [3] and [4]. Software version 1.0 was built based on that design and released in February 2006. The goal of that initial release was to evaluate proposed solutions and interfaces and to demonstrate a system supporting the full functional cycle involving interactions between applications and Lambda Station, Lambda Station and the site LAN, and pairs of Lambda Stations synchronizing network configurations at their sites. The services implemented in software version 1.0 are accessible via SOAP, however no great efforts were made yet for interoperability across heterogeneous Web Services platforms.

The initial design of Lambda Station created challenging requirements for underlying implementation. In order to build an interoperable decentralized system, we decided to employ a Service Oriented Architecture (SOA) approach. The Lambda Station in that case would be built as an orchestrated composition of loosely coupled services with message flow strongly defined by XML schemata. That could be achieved by utilizing the web services and XML APIs provided by each programming language we decided to support – Java, Perl and Python. For Java, we adopted the JClarens [6] framework as a convenient grid-aware toolkit. JClarens is implemented as a container on top of the open source Apache Axis [12] web services platform and provides authorization, access control and discovery services as well as SOAP messaging secured by transport layer for all Lambda Station (LS) services. The core of authentication is based on the gLite [7] security library and supports Standard Grid proxies or KCA-issued certificates to establish user connections to LS services, while authentication between Lambda Stations is based on Grid host certificates. The client interface to LS is being implemented with secure document/literal wrapped SOAP messages following recommendations of the Web Services-Interoperability Profile [8]. The document/literal format means that every message is sent as a validated XML document inside of a SOAP envelope.

Lambda Station API

To request a flow-based path, applications and remote Lambda Stations are provided several API calls [4], including:

- `openServiceTicket`

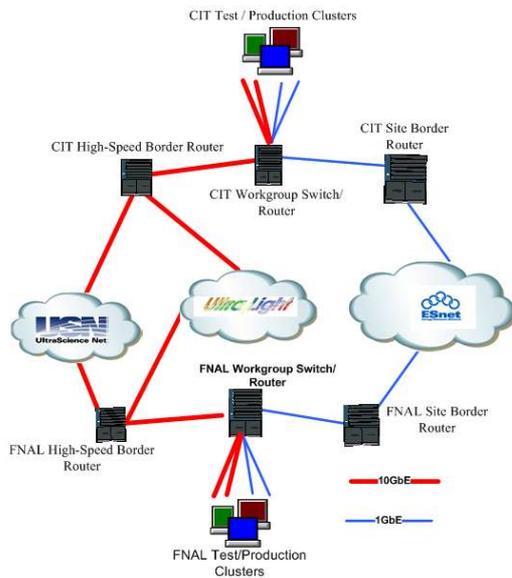


Figure 2: A Lambda Station Wide Area Testbed

- cancelTicket
 - completeTicket
 - getTicket
 - getTicketStatus
 - getFlowSpecification
 - getKnownLambdaStations
 - getKnownPBRClients
- and many others.

The detailed description of all API calls is out of scope of this paper. However, we would like to give some overview of the most important API function, `openServiceTicket`, which is used by applications and remote Lambda Stations to request an alternative network path. First we need to give two definitions, “PBR client” and “flow,” although the latter term was used above with its common meaning. In this paper, *flow* is a stream of IP packets with some attributes in common such as endpoint IP addresses (or address ranges), protocols, ports and differentiated services code point (DSCP). Any combination of these attributes can be used to identify a flow, and addresses and ports may be specified by CIDR blocks and ranges. Lambda Station is capable of dynamically reconfiguring local network infrastructure for PBR routing based on these attributes. Lambda Station controls a network path for PBR clients. A **PBR client** is an entity residing on one or more end system that generates flows that could be subjected to policy based routing. End systems sourcing and sinking traffic need to be connected to a PBR capable network infrastructure.

PBR clients are identified or created by cooperation of system and network administrators and defined in terms of flow attributes. Hence, multiple PBR clients can be defined on the same set of end systems. Lambda Station identifies PBR clients by site-wide unique identifiers. Combined with a site identifier, it identifies a PBR client globally. Predefined PBR client's information or more specific information provided in an `openServiceTicket` request allows Lambda Station to decide what parts of the local area network need to be reconfigured.

There are several different scenarios of how Lambda Station controls selective forwarding of traffic. In the simplest case, an application or a remote Lambda Station places an `openServiceTicket` request, and specifies source and destination PBR clients, desired bandwidth, boarding (a time when Lambda Station can begin configuring the network), start and end times for data movement. A unique ID will be returned immediately in response to an authenticated and authorized `openServiceTicket` request. This ID can be used by applications to track the status of path provisioning, getting additional information needed for flow marking, e.g. DSCP assigned by remote Lambda Station to the corresponding ticket at its end, as well as to synchronize actions with the remote site if, for example, the remote application cancels the ticket.

Many data movement applications, for example Storage Resource Manager [10,11] get requests to move or to schedule movement of additional files “on the fly” and may initiate an additional `openServiceTicket` call. If the flow parameters in the new call fall within those of an existing ticket, Lambda Station does not need to repeat all negotiations and network configuring. It will return the ID of an already existing ticket, possibly with an extension of its end time. This is the **Join** mode of `OpenServiceTicket`. Configurable authorization and quota parameters govern extension of existing tickets.

The `OpenServiceTicket` API call relies on pre-defined PBR clients at both ends because it tells Lambda Stations what network devices need to be reconfigured. At this time such information can not be automatically discovered. How can an application know the names of these clients? There are several ways to provide this information for applications. In the first, if the application is capable of invoking other Lambda Station services it can ask the local site's Lambda Station for the information (with `getKnownLambdaStations`,

getKnownPBRClients, ipToPBRClient). The second way is to add this information into the application's specific configuration files. And, finally, openServiceTicket allows specification of source and/or destination addresses of the systems involved in data transfers rather than their PBR client names. The site Lambda Station will try automatically to determine corresponding PBR clients at both sites to use it for network configuring.

DSCP Tagging.

Provisioning of alternate paths involves generating requests for service, negotiating parameters with the remote site, configuring local and wide area networks and marking specific flows. Obviously it takes some time to prepare the networks. Lambda Station software version 1.0 is capable of completing all these steps including dynamic reconfiguring of networks within 3 to 5 minutes. Many applications use ephemeral transport ports that are not known before a connection is opened. They may also change dynamically during a session. Therefore it is desirable, but not strictly necessary, to know the criteria for selecting flows before data transfer begins. A DSCP value is one of a few keys that can be specified in advance. A Lambda Station design does not require DSCP but can use it when available.

Although DSCP can help solve the problem of defining a flow prior to the start of data transfer, it also introduces additional complexity. First, preservation of DSCP is not guaranteed in the WAN. Second, for dynamically configurable networks DSCP tagging needs to be synchronized between sites and depends on the status of their networks. At this time, Lambda Station software does support two different modes to work with DSCP. In the first mode, a site may choose to use fixed DSCP values to identify all traffic that will be switched by Lambda Station. Lambda Station then advises applications when to apply that DSCP value, and router configurations remain constant. This mode will typically be used by sites that do not want their network devices dynamically reconfigured under Lambda Station's control.

In the second mode, a DSCP value is assigned on a per ticket basis by the local Lambda Station. The same DSCP code can be used by multiple tickets as long as the source and/or destination IP addresses are used as additional flow selectors.

Authorization and Authentication

A Lambda Station relies on the authentication schemes of the operating environment and frameworks used to integrate its components. The current Lambda Station v1.0 software uses basic (password) authentication over SSL or X.509 client and host certificates. Version 2 is being implemented in java based on gLite[7] security libraries.

Authorization rules control access to certain functions based on the identity of the requester. Three privileges are defined:

- new ticket operations (alias `new`) allow the requester to create, complete, cancel and modify tickets
- join mode operations (alias `join`) allow joining new requests to existing ticket.
- extension mode allows joining to an existing unexpired ticket and extending the active time of the original ticket.

Resource Monitoring

The final objective of provisioning an alternate path for selective flow is to increase overall performance of data movement. Achieving high data transfer rates depends on many factors. Researching aspects of high performance transport is not a goal of this project. However, when we steer selected flows onto an alternate, high bandwidth path, the user expects increased performance. Even advanced R&D networks are finite. That is why Lambda Station controls a site's use of high impact networks to avoid assigning too many tickets on the same links. At this time monitoring of resources is based on bandwidth requested via openServiceTicket call (or assigned by default). Determination of true available bandwidth by network monitoring is not yet integrated. In the future, we plan to add real-time monitoring and short-term forecasting capabilities to the Lambda Station Resource Allocation and Monitoring module.

Network Configuration

Lambda Station *deals with the last-mile problem in local networks*. It provides the means to adapt production network facilities to support access to advanced and/or research networks. At this time, Policy Based Routing is chosen as the technology for selective flow based forwarding. PBR rules are created

dynamically on-demand of applications, and applied within the LAN on work group, core and border routers. Configuring PBR rules involves the completion of several tasks including creating route map statements, applying them to appropriate interfaces and creating access control lists to match traffic. At the current stage of the project, we are using statically pre-configured route map statements applied to the interfaces. However, extended

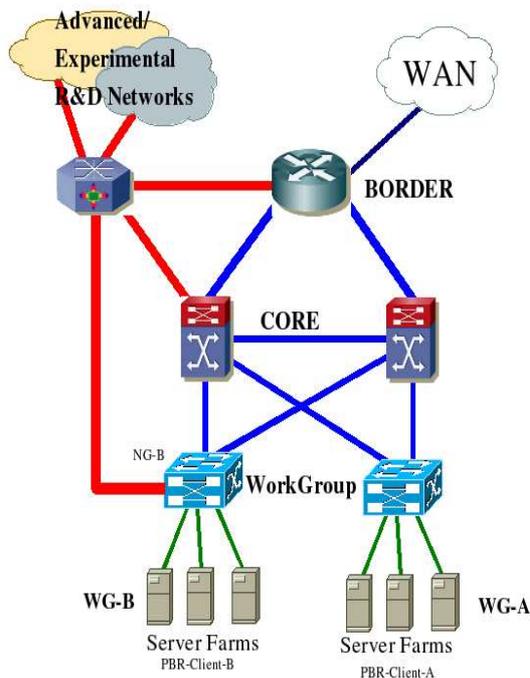


Figure 3: A hierarchical network model

access control lists can be created dynamically based on flow match criteria provided in the application's requests.

Typically, a campus network can be presented by a hierarchical design with several logical layers. Such a hierarchical layout for a work group based approach to build campus networks is depicted in figure 3. It consists of work group, core and border layers. Depending on site's specific network structure, access to R&D networks for different user groups may need to be configured at several layers. For the architecture in figure 3, outbound traffic of WG-B can be switched to R&D networks at the work group layer because it has a direct connection to the R&D admission devices. In order to get incoming traffic from R&D networks forwarded via a symmetric path, the inbound route for WG-B needs to be configured at the R&D layer. WG-A has no direct connection to R&D from its work group layer, so PBR rules must be applied at the

network core and R&D layer for both inbound and outbound traffic. **Generally speaking, work groups may require PBR rules to be applied on multiple layers of the campus network for one or both directions of traffic.**

Lambda Station does not need to deal with such architecture rather than use more simple logical grouping based on the same set of policy rules (Figure 4). Components of that model are PBR-clients, groups of network devices and multiple external network connections. Let us assume that there are several alternate wide-area networks available to a site. In figure 4 the drawings in blue represent the regular production network topology. In green and red are alternative R&D Networks with perhaps higher bandwidth available but not intended for production or commodity use. The NG-A, NG-B and NG-C are network group devices connecting correspondent PBR clients. In figure 4, it will be necessary to apply RED rules to NG-A workgroup devices and NG-ADM border group in order for nodes in network A to reach the red topology. This is because there is a direct connection from NG-A to the admission point of the RED topology. However, to access the GREEN topology, Lambda Station needs to reconfigure workgroup NG-A, NG-C network core devices and the NG-ADM border devices.

The goal of Lambda Station is to forward traffic of PBR-Clients, designated down to per-flow granularity, toward the alternate networks, on demand from applications. In order to accomplish that goal Lambda Station will need to reconfigure one or several groups of devices with a set of rules for one or both directions of traffic. Possibly different sets of rules will be applied to different groups of devices. How to group these devices depends on the site network design and involves considering physical topology of the network and a need to minimize management efforts. For example, if a network administrator can reduce the number of rules or use the same set of rules for all work groups on several network layers it will certainly simplify management. As long as the same PBR rules are applied on several layers of hierarchical work group architecture Lambda Station network model can be represented by only one group of devices.

NETWORK AND LAMBDA STATION AWARE APPLICATIONS

In the case of selective flow based forwarding, a network and host system may both be involved in the forwarding decision. Thus applications need to be aware of the network, instant status and current capabilities. If the application is designed to exploit advanced R&D networks it needs to be aware of Lambda Station service and be able to interact with the site Lambda Station to acquire the necessary information.

Lambda Station awareness (LS-awareness) is the capability in an application to request Lambda Station service. In addition to interfacing to the Lambda Station server, this may mean marking the DSCP values in packets appropriately for a service. It may also mean communicating additional information between local and remote applications.

Isiperf – a sample Lambda Station aware application

As an example of a Lambda Station aware application

request for an alternate path and watches its progress. If the path is established it starts DSCP marking of iperf's packets as requested. It also performs some other actions related to the ticket's status. For example, if the ticket is cancelled it will stop tagging.

A Lambda Station aware Storage Resource Manager

Storage Resource Manager (SRM)[11] provides access to storage elements distributed over a variety of storage systems in the grid architecture. It specifies a unified interface for initiating data transfer between heterogeneous storage systems. Fermilab's SRM implementation has been modified to invoke Lambda Station to set up policy based routing and reserve network paths for data transfer. The use of Lambda Station is controlled by a new SRM configuration parameter, and a new file defines the mapping between data URLs and PBR clients. Modifications, including enabling and disabling use of Lambda Station, can be made without restarting the SRM server. During file copy requests SRM server sends a request to the local Lambda

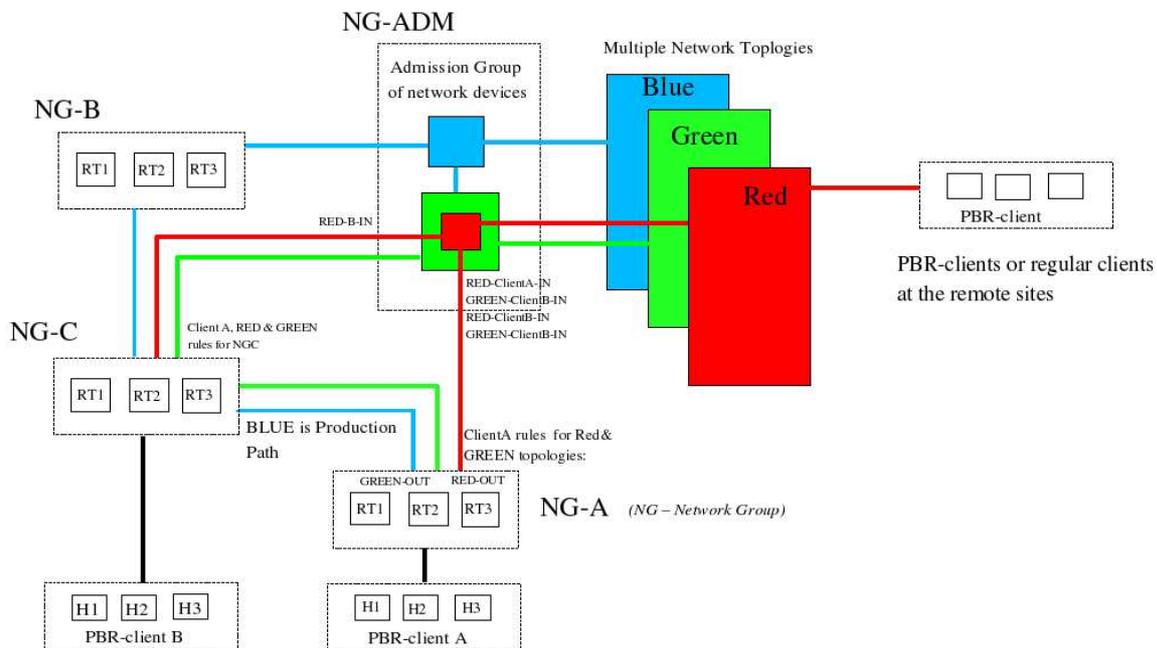


Figure 4: A Lambda Station logical groups network model

we developed a wrapper for the well-known iperf network performance measurement tool. The Isiperf starts iperf as usual. In the background it initiates a Lambda Station client process which places a ticket

Station for a data transfer path. The local Lambda Station communicates with the remote Station to resolve the path. If the path can be established fitting the parameters of the request, the requesting SRM server gets a ticket from the

local Lambda Station with several parameters describing reserved resources. Currently SRM server relies on the end time information to assess whether the reservation time is enough for transferring data. Knowing the size of data to be transferred and an estimate of transfer rates, the SRM server estimates transfer time and, if needed, requests extension of the end time of the ticket. Lambda Station aware SRM servers exist in dCache clusters at Fermilab's Feynman Advanced Projects Laboratory (FAPL) and at Caltech's CMS Tier-2 centre for development and test purposes. The FAPL dCache cluster runs two SRM servers on different TCP ports. One of them is standard and another is Lambda Station aware, this demonstrating a low-risk migration path.

THE RESULTS OF FLOW BASED ON-DEMAND ROUTING

The current software version was used to build a Lambda Station testbed to evaluate a number of network aware applications between Fermilab and Caltech. The

paths with different characteristics. One can see the TCP sending rate ramping up in the usual way when the traffic is shifted from a congested ESNET tail circuit (red) to UltraScience Net (blue). Then there is a sharper rise in the throughput when Path MTU Discovery finds that jumbo frames are supported on the alternate path. Other tests as proof of concept were done at SuperComputing 2005 and demonstrated flow based switching between SCinet and Fermilab [5].

SUMMARY

The current status of Lambda Station project provides sufficient results to anticipate a production quality system interfacing storage and computing facilities with Advanced R&D Networks. The capability of Lambda Stations to complete all negotiations and site's network configuration within 3 to 5 minutes upon receiving requests from applications is considered tolerable because applications need not wait for completion of Lambda Station procedures. While negotiations are in progress

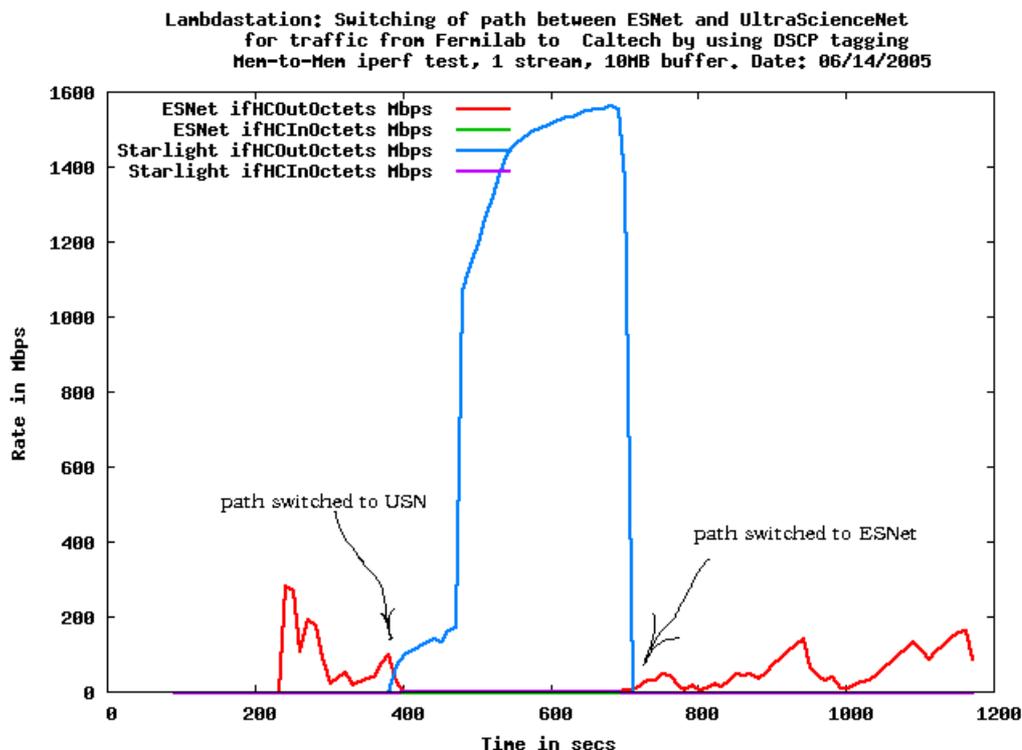


Figure 5: Selective flow switching onto two different paths

ESNet was our production path and two high bandwidth networks, UltraScienceNet and UltraLight were used as alternative network topologies. The graphs in Figure 5 demonstrate typical behaviour of switched flows on two

traffic will go by the regular path.

Experience with Lambda Station to use different applications has also demonstrated that there are still a lot issues that need to be worked out. Fully utilizing

Lambda Station capabilities makes it desirable to have network awareness capabilities in applications. It introduces a significant level of complexity.

However, in our view the Lambda Station project is based on a long term perspective driven by the increasing need to operate networks in a world with dynamically provisioned optical paths, diverse local network infrastructures and a great number of end-nodes at each facility.

REFERENCES

- [1] Lambda Station Project Web site
<http://www.lambdastation.org/>
- [2] Donald L. Petravick, Fermilab, *LambdaStation: Exploring Advanced Networks in Data Intensive High Energy Physics Applications*, Project Proposal, <http://www.lambdastation.org/omnibus-text.pdf>
- [3] Phil DeMar, Donald L. Petravick. *LambdaStation: A forwarding and admission control service to interface production network facilities with advanced research network paths*, Proceedings of CHEP2004, Interlaken, Switzerland, 27th September - 1st October 2004.
- [4] A.Bobyshev, M.Crawford et al., Lambda Station: Production applications exploiting advanced networks in data intensive high energy physics, Proceedings of CHEP06, TIFR, Mumbai, India, 13-17 February 2006.
- [5] A.Bobyshev, M.Crawford, V.Grivalinus, M.Grigoriev, R.Rechenmacher. Investigating the behavior of network aware applications with flow-based path selection, Proceedings of CHEP06, TIFR, Mumbai, India, 13-17 February 2006
- [6] M. Thomas, C. Steenberg et al., "JClarens: A Java Framework for Developing and Deploying Web Services for Grid Computing," ICWS, pp. 141-148, IEEE International Conference on Web Services (ICWS'05), Orlando, FL, 2005.
- [7] EGEE Global Security Architecture, EU Deliverable DJRA3.1,EGEE-JRA3-TEC-487004-DJRA3.1-v-1.1, <http://edms.cern.ch/document/487004/>
- [8] *WS-I Basic Profile Version 1.1*, Final Material, 2004-08-24, Editors: K. Ballinger(Microsoft), D. Ehnebuske(IBM), et al., the Web Services-Interoperability Organization.
- [9] GARA - Globus Architecture for Reservation and Allocation (grid middleware), <http://www.globus.org/research/resource-management.html>
- [10] Fermilab mass storage including dCache, <http://grid.fnal.gov/>
- [11] Fermilab SRM Project, <https://srm.fnal.gov/twiki/bin/view/Main/WebHome>
- [12] Apache Web Services Project, <http://ws.apache.org/>